IoT Platform using Geode and ActiveMQ

Swapnil Bawaskar
@sbawaskar
sbawaskar@apache.org

Scalable IoT Platform
Agenda

• Introduction
  • IoT
  • MQTT
• Apache ActiveMQ Artemis
• Apache Geode
• Real world use case
• Q&A
IoT

- Devices collect and send data to brokers
- Clients process data to deliver business value
- IoT data platform considerations
  - Protocol
  - How to read
  - How to Analyze
  - How to scale
Protocol

- MQTT
  - Message Queuing Telemetry Transport
  - Based on TCP/IP
  - Optimized binary protocol
  - No type system
  - Provides different QoS levels
  - Low energy consumption
ActiveMQ Artemis

- Subproject of ActiveMQ
- Non blocking architecture
  - High Performance
  - Multi Protocol
  - Embeddable
  - Clustered
- Persistence
  - Journaled
  - Relational database
When dealing with large number of devices
Scaling

- Brokers form cluster
- Clients are load balanced
Scaling

• Need to scale the processors
Scaling

• Processors do not see all data
Scaling

Figure 3.1 Artemis High Level Architecture
Scaling

Figure 3.1 Artemis High Level Architecture
What is it?
A distributed, memory-based data management platform for data oriented apps that need:

• high performance, scalability, resiliency and continuous availability
• fast access to critical data set
• location aware distributed data processing
• event driven data architecture
## Numbers Everyone Should Know

<table>
<thead>
<tr>
<th>Operation</th>
<th>Time (ns)</th>
<th>Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1 cache reference</td>
<td>0.5</td>
<td>0.0005</td>
</tr>
<tr>
<td>Branch mispredict</td>
<td>5</td>
<td>0.005</td>
</tr>
<tr>
<td>L2 cache reference</td>
<td>7</td>
<td>0.007</td>
</tr>
<tr>
<td>Mutex lock/unlock</td>
<td>100</td>
<td>0.1</td>
</tr>
<tr>
<td>Main memory reference</td>
<td>100</td>
<td>0.1</td>
</tr>
<tr>
<td>Compress 1K bytes with Zippy</td>
<td>10,000</td>
<td>0.01</td>
</tr>
<tr>
<td>Send 1K bytes over 1 Gbps network</td>
<td>10,000</td>
<td>0.01</td>
</tr>
<tr>
<td>Read 1 MB sequentially from memory</td>
<td>250,000</td>
<td>0.25</td>
</tr>
<tr>
<td>Round trip within same datacenter</td>
<td>500,000</td>
<td>0.5</td>
</tr>
<tr>
<td>Disk seek</td>
<td>10,000,000</td>
<td>10</td>
</tr>
<tr>
<td>Read 1 MB sequentially from network</td>
<td>10,000,000</td>
<td>10</td>
</tr>
<tr>
<td>Read 1 MB sequentially from disk</td>
<td>30,000,000</td>
<td>30</td>
</tr>
<tr>
<td>Send packet CA-&gt;Netherlands-&gt;CA</td>
<td>150,000,000</td>
<td>1.5</td>
</tr>
</tbody>
</table>

Who are the users?

- 17 billion records in memory
  - GE Power & Water's Remote Monitoring & Diagnostics Center
- 3 TB operational data in-memory, 400 TB archived
  - China Railways
- 4.6 Million transactions a day / 40K transactions a second
  - China Railways
- 120,000 Concurrent Users
  - Indian Railways
China Railway Corporation

Population: 1,401,586,609

Indian Railways

Population: 1,251,695,616

World: ~7,349,000,000

~36% of the world population
- Distributed key-value store
  (java.util.concurrent.ConcurrentMap)
- Region due to old JSR-107 spec
- Both Keys as well as Values can be domain objects
Functions

- Deploy Function on all servers
- Runs in-process with the servers
Functions

• Deploy Function on all servers
• Runs in-process with the servers
Object Query Language (OQL)

Similar to SQL

- SELECT DISTINCT * FROM /exampleRegion WHERE status = 'active'

You can drill down into domain objects

- SELECT p.name FROM /person p WHERE p.pet.type='dino'

You can also invoke methods on your domain objects

- SELECT DISTINCT * FROM /person p WHERE p.children.size >= 2

Joins Possible

- Between Replicate regions

- Between one Partitioned and Replicate regions
  
  SELECT portfolio1.ID, portfolio2.status FROM /exampleRegion portfolio1, /exampleRegion2 portfolio2 WHERE portfolio1.status = portfolio2.status
Continuous Query

- Enables event-driven apps
- Register a Query with the server
  - `SELECT * FROM /tradeOrder t WHERE t.symbol='VMW' AND t.price > 100.00`
- The server then notifies when the query condition is met
  - Client implements the `CqListener` callback
- HA support
- Domain objects **not** required on the server’s class-path
Fixed or flexible schema?

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>age</th>
<th>pet_id</th>
</tr>
</thead>
</table>

or

```json
{
  id : 1,
  name : "Fred",
  age : 42,
  pet : {
    name : "Barney",
    type : "dino"
  }
}
```
C#, C++, Java, JSON

Portable Data eXchange

<table>
<thead>
<tr>
<th>pdx</th>
<th>length</th>
<th>dsid</th>
<th>typeid</th>
<th>fields</th>
<th>offsets</th>
</tr>
</thead>
</table>

No IDL, no schemas, no hand-coding
Schema evolution (Forward and Backward Compatible)

* domain object classes not required
No need to bring down cluster when domain objects change
Efficient for queries

```
{  
id : 1,
name : "Fred",
age : 42,
pet : {
    name : "Barney",
    type : "dino"
}
}
```

```
SELECT p.name FROM /Person p WHERE p.pet.type = "dino"
```
But how fast is it?

Benchmark: https://github.com/eishay/jvm-serializers
Schema evolution

v2 objects preserve data from missing fields

v1 objects use default values to fill in new fields

PDX provides forwards and backwards compatibility, no code required
• Telemetry data from machines
• Predicting failure
  • Outside one standard deviation
  • Evaluating markov model
• Use functions to iterate over data
• Use CQs to notify
• Update CQs based on function results
Questions?